Chapter 14

Creating More Complex Database Applications

Objectives

In this chapter, you will:

• Implement a one-to-one association in a database application
• Implement a one-to-many association in a database application
• Apply parent-child (hierarchical) dataset relationships
• Implement an association class in a database application

Implementing a One-to-One Association in a Database Application

• Relationships between Customer and Boat
  – Mandatory
  – One-to-one relationships in both directions
  • A customer always owns exactly one boat
  • A boat is always owned by exactly one customer

Understanding the CustomerAndBoat Database

• Tables of CustomerAndBoatDatabase
  – BoatTable
  – CustomerTable

Implementing a One-to-One Association in a Database Application

Understanding the CustomerAndBoat Database

• BoatTable fields
  – StateRegistrationNo
  – BoatLength
  – Manufacturer
  – Year
  – CustomerPhoneNo

Figure 14-3: The CustomerAndBoatDatabase window

Figure 14-4: Contents of BoatTable
Understanding the CustomerAndBoat Database

- CustomerTable fields
  - Name
  - Address
  - PhoneNo

![Contents of CustomerTable](image)

Understanding the CustomerAndBoat Database

- Primary key
  - A field that uniquely identifies each record in a relational database table
  - CustomerTable primary key: PhoneNo
  - BoatTable primary key: StateRegistrationNo

- Foreign key
  - A field in one table that serves as a primary key in a related table
  - BoatTable foreign key: CustomerPhoneNo

Using SQL to Join Tables in the Database

- Joining two tables in SQL
  - Using foreign and primary keys to link information in one table to information in another table

- Example
  - Find and display the state registration number and manufacturer of all boats in BoatTable, with the name, address, and phone number of each boat’s owner

  ```sql
  SELECT StateRegistrationNo, Manufacturer, Name, Address, PhoneNo
  FROM BoatTable, CustomerTable
  WHERE CustomerPhoneNo = PhoneNo
  ```

- Example:
  - Display the name and address of the owner of the boat with state registration number MO98765

  ```sql
  SELECT StateRegistrationNo, Name, Address
  FROM BoatTable, CustomerTable
  WHERE CustomerPhoneNo = PhoneNo
  AND StateRegistrationNo = 'MO98765'
  ```

Establishing a Common Connection to the Database

- CustomerAndBoatDatabaseConnect program
  - Establishes a common connection to CustomerAndBoatDatabase
  - Used by
    - CustomerDA
    - BoatDA
Modifying the Customer Class

- Additions to the Customer class to reflect the link between CustomerTable and BoatTable
  - A boat reference attribute
    - Enables a customer instance to know about its associated boat instance
    - Initialized to Nothing in the constructor
  - A setter method to set the boat reference attribute
  - A getter method to retrieve the boat reference attribute

Introducing the BoatDA Class

- BoatDA class
  - Needed because the CustomerAndBoatDatabase application must
    - Find
    - Add
    - Delete
    - Update
    - records in BoatTable

Introducing the BoatDA Class

- Methods of the BoatDA class
  - Initialize
    - Opens a connection to the database
  - Terminate
    - Closes the connection to the database
  - Find
    - Defines an SQL SELECT statement that retrieves a particular record from BoatTable

Introducing the BoatDA Class

- Methods of the BoatDA class (continued)
  - GetAll
    - Similar to the Find method, but returns an ArrayList of boat references
  - AddNew
    - Uses the SQL INSERT INTO clause to insert the information for a new boat into BoatTable
  - Update
    - Used to change the value of a field or fields of a boat
  - Delete
    - Deletes a boat record from the table

Modifying the Boat Class to Work With BoatDA

- Additions to the Boat class needed to work with the BoatDA class
  - Four shared methods to invoke the following methods of BoatDA
    - Initialize
    - Find
    - GetAll
    - Terminate

Modifying the Boat Class to Work With BoatDA

- Additions to the Boat class needed to work with the BoatDA class (continued)
  - Three instance methods to invoke the following methods of BoatDA
    - AddNew
    - Update
    - Delete
  - Revision of the TellAboutSelf method
    - To improve readability of the information returned to the calling program
Modifying the CustomerDA Class

- Modifications in CustomerDA
  - Needed to support joining the information from BoatTable and CustomerTable
  - A boat reference variable and variables to represent boat attributes
  - Find method
    - Extended to retrieve data from both tables

- Modifications in CustomerDA (continued)
  - AddNew method
    - When a customer record is inserted
      - Invokes the AddNew method of the BoatDA class to insert the associated boat record into BoatTable
  - Delete method
    - When a customer record is deleted
      - The corresponding boat record is also deleted

Implementing a One-to-Many Association in a Database Application

- The relationship between Dock and Slip
  - A one-to-many relationship
  - A dock contains many slips
  - A slip belongs to exactly one dock

Understanding the Tables in DockAndSlipDatabase

- Tables of the DockAndSlipDatabase
  - DockTable
  - SlipTable
- Columns of DockTable
  - DockId (the primary key)
  - Location
  - Electricity
  - Water

- Columns of SlipTable
  - SlipNo
  - DockId
  - Width
  - SlipLength
  - BoatId

- SlipNo and DockId
  - Together uniquely identify a slip
  - Form a concatenated key
    - Concatenated key: a key that contains more than one field (or column) in the database
Understanding the Tables in DockAndSlipDatabase

- DockId in SlipTable is a foreign key to information in DockTable

Establishing a Common Connection to DockAndSlipDatabase

- DockAndSlipDatabaseConnect program
  - Establishes a single connection to DockAndSlipDatabase
  - Shared by the PD and DA classes that require access to the database
  - Identical to CustomerAndBoatDatabaseConnect, except for the specified data source name

Modifying the Dock and Slip Classes

- The Dock class (now called mDock)
  - Modifications to the Dock class
    - Class name changed to mDock
      - To eliminate conflict with VB.NET’s Dock class (used for Windows forms)
    - Addition of the System.Data.OleDb namespace
    - Addition of a TellAboutSelf method
    - Does not currently require methods to
      - Insert records
      - Update records
      - Delete records

- Slip class
  - No need to modify the Slip class
  - Already includes the code to associate a slip with its dock
  - Does not need
    - Insert method
    - Delete method
    - Update method
  - Does not need Find and GetAll methods of its own
  - A SlipDA class is not needed for the DockAndSlipDatabase application

Introducing the DockDA Class

- DockDA class
  - Used to access information in DockTable
  - Does not require the following methods
    - Insert
    - Delete
    - Update
  - Find method
    - Defines the SQL SELECT statement needed to extract dock and slip information from the database

- DockDA class (continued)
  - GetAll method
    - The SQL query returns dock and slip information for all docks and slips in the marina, sorted in order by dock and then by slip
    - Uses control-break logic
      - A control-break occurs when there is a change in the value of a variable that is used to group a list of sorted items
Applying Parent-Child (Hierarchical) Dataset Relationships

- An application is best represented by a hierarchical, or parent-child, relationship of tables
- The dock and slip relationship can be a parent-child relationship
  - A dock can have many slips
  - Each slip must be associated with only one dock

Processing logic for a parent-child relationship between DockTable and SlipTable
- For each row of DockTable
  - Iterate over SlipTable finding all the slips that belonged to the corresponding dock

VB .NET datasets provide the functionality of establishing parent-child relationships of the tables in a dataset
- For example:
  - GetAll method of the DockDA class can be rewritten using a parent-child relationship of a dataset

Understanding the Tables in CustomerLeaseSlipDatabase

- Database tables of the application
  - CustomerTable
  - LeaseTable
  - SlipTable
Understanding the Tables in CustomerLeaseSlipDatabase

• LeaseTable
  – Customer phone number: a foreign key to link to CustomerTable
    • A customer’s phone number
      – Used as the primary key for LeaseTable (CustomerPhoneNo)
      – Used as the primary key for CustomerTable (PhoneNo)
    – Columns for slip number and dock ID serve as a concatenated foreign key to records in SlipTable

Understanding the Tables in CustomerLeaseSlipDatabase

• CustomerLeaseSlip application has
  – Four problem domain classes
    • Customer
    • Slip
    • Lease
    • AnnualLease
  – Three data access classes
    • CustomerDA
    • SlipDA
    • LeaseDA
  – A test class
  – A class to establish a connection to the database

Establishing a Connection to CustomerLeaseSlipDatabase

• CustomerLeaseSlipConnect program
  – Establishes a single connection to CustomerLeaseSlipDatabase
  – Shared by the PD and DA classes involved in this application
  – Identical to the other connect programs, except for the specified data source name

Modifying the Customer Class

• Modifications needed in the Customer PD class
  – A lease reference of the AnnualLease type
    • Initialized to nothing by the constructor
  – A setter method to set the AnnualLease reference variable
  – A getter method to retrieve the AnnualLease reference variable

Modifying the Lease and AnnualLease Classes

• Lease class
  – Must associate a lease with its customers and slips
  – Additions needed to accomplish this
    • A slip reference variable
      – Initially set to Nothing in the constructor
    • A customer reference variable
      – Initially set to Nothing in the constructor
    • Getter methods to retrieve the slip and customer variables
    • Setter methods to set the slip and customer variables
Modifying the Lease and AnnualLease Classes

- Additions to the AnnualLease subclass
  - Initialize method
  - Terminate method
  - Find method
  - AddNew method

Modifying the Slip Class

- Slip class
  - Must associate a slip with its corresponding lease and customer
  - Additions needed to accomplish this
    - An AnnualLease reference
      - Initially set to Nothing in the constructor
    - Initialize method
    - Terminate method
    - Find method

Modifying the Slip Class

- Slip class
  - Additions needed to associate a slip with its corresponding lease and customer (continued)
    - LeaseAnnualSlip method
      - Associates the slip with its corresponding lease instance

Introducing the SlipDA Class

- SlipDA class
  - Needed in the CustomerLeaseSlip application
  - Contains
    - Variables for
      - Slip attributes
        - Establishing the database connection
    - Initialize method
      - Identical to those in the other DA classes

Introducing the SlipDA Class

- SlipDA class (continued)
  - Terminate method
    - Identical to those in the other DA classes
  - Find method
    - Defines a SQL SELECT statement that returns information from SlipTable for a particular slip

Introducing the AnnualLeaseDA Class

- AnnualLeaseDA class
  - Needed in the CustomerLeaseSlip application
    - Information about annual leases must be found and inserted in the database
  - Contains
    - Variables for AnnualLease attributes
    - Initialize method
    - Terminate method
Introducing the AnnualLeaseDA Class

- AnnualLeaseDA class contains (continued)
  - Find method
    - Requires an SQL query that retrieves data from CustomerTable, LeaseTable, and SlipTable
  - AddNew method
    - Adds a lease record to the database

Summary

- A primary key is an attribute (or combination of attributes) that uniquely identifies a single record in a relational database table
- A foreign key is an attribute (or column) in one relational database table that serves as a primary key in a different (or foreign) table
- A concatenated key (primary or foreign) is one that consists of more than one attribute (or column) in the database table

- When multiple tables are involved, DA Insert and Delete methods must incorporate measures to preserve the integrity of the database
- A control-break occurs when there is a change in the value of a variable that is used to group a list of sorted items
- VB .NET datasets provide the functionality to visualize and process data hierarchically, referred to as a parent-child relationship